ELASTIC NET MODEL

#vehicle data

#saimohan

#FIAT500 DATA

In [156]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [157]:
```
data=pd.read_csv("/home/placement/Desktop/saimohan data/csv files/fiat500.csv")
```

In [183]:
```
data.describe()
```

Out[183]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [184]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [185]:
```
data.tail(5)
```

Out[185]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [158]:
```
z=data1.loc[(data1.previous_owners==1)]
```

In [182]:
```
z
```

Out[182]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1389 rows × 8 columns

In [159]:
```
data.head(5)
```

Out[159]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700

In [160]:
```
data1=data.drop(["lat","lon","ID"],axis=1)
```

In [161]:
```
#string--num
data1=pd.get_dummies(data1)
```

In [162]:
```
data1
```

Out[162]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

In [163]:
```
#z=data1.loc[(data.model=="lounge")]
```

In [164]:
```
z
```

Out[164]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1389 rows × 8 columns

In [165]:
```
#which  the parameter is predected values can be removed from the data file
#1) we copied the data into another data("y")
#2) later we can removed those file from main data set
y=z['price']
x=z.drop(['price'],axis=1)
```

In [166]:
```
#i am calling function to split
#split enter data into ->67% traning  ,  ->33% testing

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=42)
```

In [167]:
```
x_train.head(5)
#testing and training
```

Out[167]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
956	51	790	26210	1	1	0	0
1411	51	1461	46108	1	1	0	0
333	51	456	26526	1	1	0	0
1452	51	1247	75000	1	1	0	0
1369	51	701	36500	1	1	0	0

In [168]:
```
y_train.head(5)
```

Out[168]:
```
956     8750
1411    8000
333     9980
1452    8000
1369    9990
Name: price, dtype: int64
```

In [169]:
```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(x_train, y_train)
```

Out[169]:

GridSearchCV

```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

estimator: ElasticNet

ElasticNet

```
ElasticNet()
```

In [178]:
```
elastic_regressor.best_params_
```

Out[178]:
```
{'alpha': 0.01}
```

In [170]:
```
elastic=ElasticNet(alpha=.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [171]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[171]: 0.8488682857174344

In [176]:
```
from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[176]: 603966.023413073

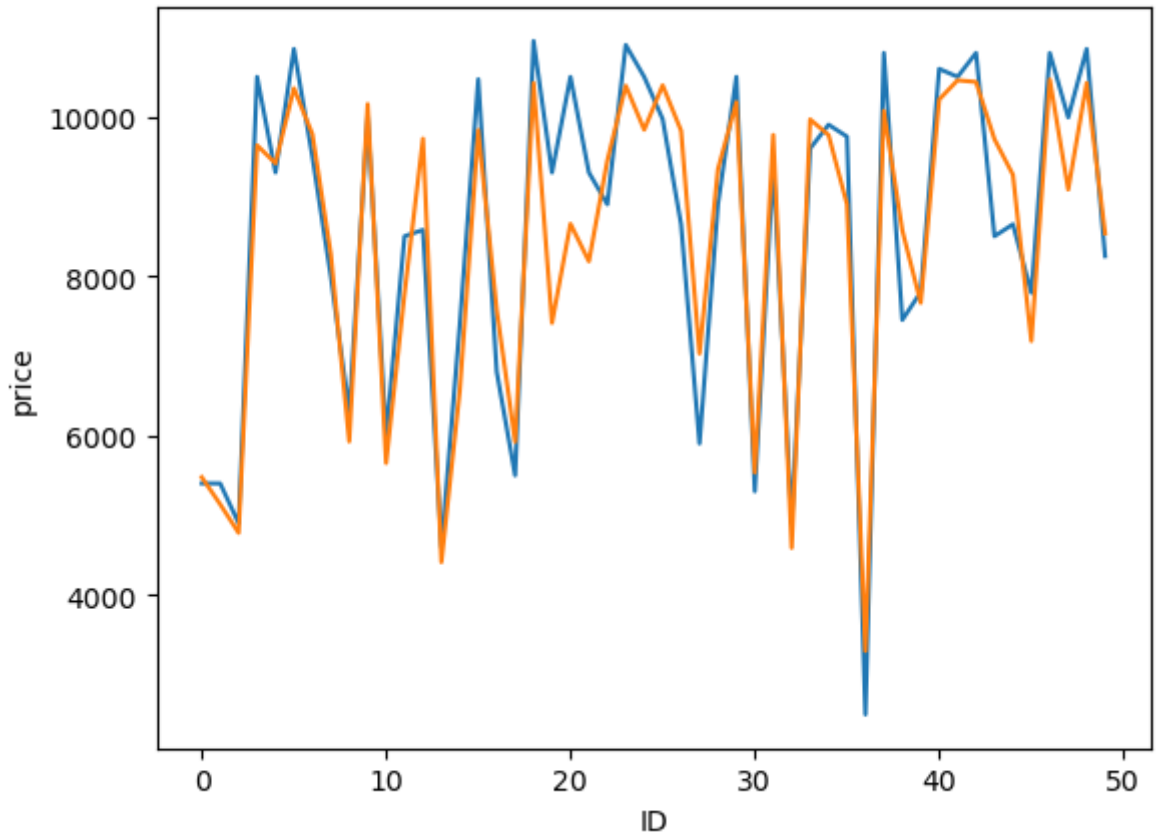
In [180]:
```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[180]:

	index	price	predicted	ID
0	625	5400	5477.052458	0
1	187	5399	5137.435504	1
2	279	4900	4778.564980	2
3	734	10500	9640.895436	3
4	315	9300	9415.174300	4
5	652	10850	10356.323449	5
6	1472	9500	9781.272728	6
7	619	7999	8276.238400	7
8	992	6300	5925.267808	8
9	1154	10000	10158.433547	9
10	757	6000	5654.915390	10
11	1299	8500	7779.899617	11
12	400	8580	9724.510940	12
13	314	4600	4411.587148	13
14	72	7400	6568.196031	14

In [181]:
```
###################################
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='price',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[181]: []



In [ ]: